STAT 672 Final Project: Stochastic Gradient Descent

Tom Wallace

April 10, 2018

1 Introduction

1.1 Organization

This paper is divided into four sections. The remainder of this **Introduction** section gives motivation for stochastic gradient descent (SGD): why do we care about it? The **Method and Theory** section presents the mathematics of SGD and some of its notable properties. The **Applications** highlights the real-world settings and uses of SGD, including a case study data analysis. The **Conclusion** section summarizes overall findings and notes some areas for further research in SGD.

1.2 Motivation

Optimization is fundamental to statistical modeling. The chief task of statistical modeling is to characterize the relationship between explanatory variables and an outcome variable, and the chief method for doing so is to estimate values for coefficients that best relate each explanatory variable to the outcome variable. The term "best" implies picking coefficient values that maximize some measure of goodness (e.g. likelihood) or minimize some measure of badness (e.g. loss function). Mathematical optimization is the typical route to achieving such minimization or maximization. Two important considerations for optimization are parametric assumptions and computational complexity. SGD, an optimization technique, is particularly motivated by these considerations.

1.2.1 Parametric vs. non-parametric

Assuming that the outcome variable follows a particular statistical distribution aids the computation of optimal coefficients. For example, assumptions in ordinary least squares (OLS) regression—assumptions that readers almost certainly are familiar with and so will not be repeated here—allow a closed form solution. Suppose we have n observations. Each observation consists of an outcome variable y_i and some explanatory variables $x_{i1}, x_{i2} \dots x_{iD}$. We thus have a vector of outcomes $\boldsymbol{Y}_{n\times 1}$ and a feature matrix $\boldsymbol{X}_{n\times D}$. All variables are real numbers: $\boldsymbol{X} \in \mathbb{R}^{n\times D}$, $\boldsymbol{Y} \in \mathbb{R}^{n\times 1}$. The goal is estimate weights $w_1 \dots w_j \dots w_D$, or $\boldsymbol{w}_{1\times D}$, that relate the explanatory variables to the outcome.[1] The normal equations give:

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

Even if a parametric model does not have a closed-form solution, the parametric assumption allows some useful optimization techniques. Consider logistic regression. The maximum likelihood estimator (MLE) approach leads to a system of D equations. This system of equations typically is numerically solved using the iterative Newton-Raphson algorithm:

$$\hat{\boldsymbol{w}}_{n+1} = \hat{\boldsymbol{w}}_n - \boldsymbol{H}^{-1}(\hat{\boldsymbol{w}}_n)\boldsymbol{J}(\hat{\boldsymbol{w}}_n)$$

$\boldsymbol{J}$ is the Jacobian (the first derivative of the log-likelihood function l with respect to each w_j) and $\boldsymbol{H}$ is the Hessian (the second derivative of l with respect to $w_j, w_{j'}$). The practicality of Newton-Raphson thus depends on whether it is convenient to find $\boldsymbol{J}$ and $\boldsymbol{H}$. This is true for logistic regression because parametric and independent-and-identically-distributed (IID) assumptions mean l is a simple sum of the log probability distribution function (PDF, in this case binomial) for each observation. We "know" (assume) the form of this PDF and so are confident that the second derivative exists and is not too onerous to calculate. It may not be true, and hence Newton-Raphson will not be practical, if the Jacobian or Hessian of the function we are trying to maximize or minimize (perhaps not l) are non-existent or cumbersome.

Non-parametric modeling often is such a case. MAYBE REVISE THIS. Consider a typical supervised classification set-up: feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times D}$ and labels $\boldsymbol{Y} \in \{-1, 1\}$, with no assumptions regarding distribution. The goal is to pick a function $f_w(\boldsymbol{X})$ from hypothesis class $\mathcal{F}$ that assigns weights to features so as to predict label. The optimal weights are those that minimize empirical risk according to some loss function L that characterizes how well (or not) $f_w(\boldsymbol{X})$ classifies in terms of discrepancy between $f_w(\boldsymbol{X}_i)$ and Y_i for a particular i.

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{1}{n}\sum_{i=1}^{n} L(f_w(\boldsymbol{X}_i), Y_i)$$

Our lack of parametric assumptions means we do not have a closed-form solution a la OLS. We also have no guarantees that the loss function L has existent or easy-to-find derivatives, and hence no guarantee that Newton-Raphson is practical.

The need to conduct optimization in such non-parametric settings is a chief motivation for gradient descent (GD), of which SGD is a variant. GD does not require any parametric assumptions. It also only requires finding the first

[1]Readers may be more familiar with $\boldsymbol{\beta}$ as the notation for coefficients. $\boldsymbol{w}$ is used here to ensure common notation with other classes of models.

derivative (Jacobian) of the function. For these reasons, (S)GD is well-suited for non-parametric settings, included supervised and unsupervised statistical learning tasks.

1.2.2 Computational complexity

How an optimization technique scales with n and D is another important consideration. It is little comfort if a method reaches the correct solution but requires an excessive amount of time to do so. "Plain" or "batch" GD requires evaluating the gradient for every single observation, every single iteration, until the algorithm converges. For example, for a dataset of $n = 10^6$ that required 25 iterations to converge, batch GD would require evaluating the gradient 25×10^6 times. This linear scaling with n can cause untenably long computation time.

SGD alleviates these computational difficulties by requiring the gradient to be evaluated for only a single randomly chosen observation per iteration. This approach means convergence is "noisier" and hence requires more iterations to converge, but each iteration is less complex to compute and so can be done faster. SGD thus scales much more favorably with n than GD, and so is particularly useful for large-n applications such as typical machine learning or big data tasks.

2 Method and Theory

2.1 Setup and Notation

2.2 Derivation

[1] [2] [3] [5] [4]

2.3 Key Properties

3 Applications

3.1 SGD and Statistical Learning

3.2 Case Study:

4 Conclusion

References

[1] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[2] Léon Bottou. "Stochastic gradient descent tricks". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[4] Andrea Dal Pozzolo et al. “Calibrating probability with undersampling for unbalanced classification”. In: *Computational Intelligence, 2015 IEEE Symposium Series on.* IEEE. 2015, pp. 159–166.

[5] Martin Zinkevich et al. “Parallelized stochastic gradient descent”. In: *Advances in neural information processing systems.* 2010, pp. 2595–2603.